**SLAJD 1**

Express.js - Framework dla aplikacji webowych

**SLAJD 2**

**Co to jest Express.js?**

Express.js to framework webowy dla Node.js. Umożliwia łatwe budowanie serwerów i API, a jego minimalistyczna, elastyczna i rozbudowywalna struktura sprawia, że jest idealny do tworzenia aplikacji RESTful.

**SLAJD 3**

**Zalety Express.js**

**Prostota:** Express.js oferuje intuicyjne API, które sprawia, że nauka i praca z nim są łatwe.

**Lekkość:** Brak złożonej konfiguracji umożliwia szybkie uruchomienie projektu.

**Elastyczność:** Middleware pozwala na łatwe zarządzanie żądaniami, co daje programistom dużą kontrolę.

**Wydajność:** Działa na Node.js, co oznacza szybkie, asynchroniczne operacje.

**Społeczność:** Istnieje bogata baza pluginów i rozszerzeń, które można wykorzystać w swoich projektach.

**SLAJD 4**

Express.js, mimo wielu zalet, ma swoje wady. Po pierwsze, brak narzuconej struktury może prowadzić do problemów w większych aplikacjach. Po drugie, zarządzanie skalą i wydajnością w bardziej rozbudowanych projektach jest trudniejsze. Dodatkowo, Express wymaga ręcznej obsługi błędów, co może być uciążliwe. Middleware, mimo że elastyczne, może być trudne do debugowania w większych aplikacjach. Brak natywnego wsparcia dla TypeScript utrudnia typowanie, a korzystanie z wielu zewnętrznych bibliotek zwiększa złożoność projektu. Mimo to, wiele z tych problemów da się rozwiązać odpowiednimi narzędziami i wzorcami projektowymi.

**SLAJD 5**

**Jak działa Express.js?**

Express.js tworzy serwer HTTP i obsługuje różne rodzaje żądań, takich jak GET, POST, PUT i DELETE. Używa tras (routes) do dopasowania URL-i do odpowiednich funkcji, a także obsługuje middleware do przetwarzania danych w żądaniach.

*Przykład*

**SLAJD 6**

**Co to jest Middleware?**

Middleware to funkcje, które mają dostęp do obiektów żądania (request) i odpowiedzi (response). Służą do przetwarzania danych przed dostarczeniem odpowiedzi. Przykłady middleware to logowanie, autoryzacja, parsowanie JSON i obsługa błędów.

**SLAJD 7**

**Routing w Express.js**

Routing to proces dopasowywania URL do odpowiedniej funkcji. Express oferuje różne metody do obsługi tras, takie jak app.get(), app.post(), app.put() i app.delete(). Dodatkowo, Express pozwala na obsługę parametrów w trasach za pomocą req.params, co ułatwia dynamiczne dopasowywanie URL-i.

*Przykład*

**SLAJD 8**

**Połączenie Express.js z MySQL**

Express.js umożliwia integrację z różnymi bazami danych, w tym MySQL. W tym celu używamy biblioteki mysql, która pozwala na łatwe połączenie między Express.js a MySQL.

**SLAJD 9**

**Express.js: Pobieranie danych z MySQL**

Za pomocą Express.js możemy pobierać dane z bazy MySQL i zwracać je w formacie JSON, co ułatwia komunikację z frontendem oraz przetwarzanie danych.

*Przykład 1*

*Przykład 2 (angular)*

**SLAJD 10**

**Podsumowanie Express.js**

Podsumowując, Express.js to potężne narzędzie do budowy backendu. Jego minimalistyczna i wydajna architektura, łatwe połączenie z bazami danych oraz integracja z frontendem sprawiają, że jest szeroko stosowany w nowoczesnych aplikacjach webowych.